2207/684502
PATENT

UNITED STATES PATENT APPLICATION
FOR

BRANCH MISPREDICTION RECOVERY USING A SIDE MEMORY

INVENTOR:

NICOLAS I. KACEVAS

PREPARED BY:

KENYON & KENYON
1500 K STREET, N.W. SUITE 700
WASHINGTON, D.C. 20005

(202)220-4200

BRANCH MISPREDICTION RECOVERY USING A SIDE MEMORY

RELATED APPLICATION

This patent application is a continuation application of, and claims priority to, U.S. Application Serial No. 09/398,102, filed September 16, 1999.

FIELD

The present invention relates to an instruction pipeline in a processor. More particularly, the present invention relates to a mispredicted path side memory for an instruction pipeline.

BACKGROUND

The rate at which a computer or other processing system can process information is often dependent on the speed at which the system processor(s) execute instructions. Therefore, increased processing may advantageously be obtained by improving the speed at which processor process instructions. Many processors, such as a microprocessor found in a computer, use an instruction pipeline to speed the processing of instructions. FIG. 1 illustrates a known architecture for such an instruction pipeline. The first stage of the pipeline includes a branch prediction unit 100 and a next Instruction Pointer (IP) logic unit 110 that select an instruction to be executed. An instruction cache 120 is accessed in the second stage of the pipeline, and the instruction moves into the third stage. The instruction moves from a third stage unit 130 to a fourth stage unit 140, and so on, before reaching a branch execution unit 150 in the execution stage. The "intermediate stages" shown in FIG. 1 imply that any number of stages can exist in a pipeline. The stages may, for example, generate instructions for an instruction decoder.

Consider, for example, the following sequence of instructions:

address X1: XXX1
JCC-Y1
XXX2
XXX3
XXX4
XXX5

address Y1: YYY1
YYY2
YYY3

In this case, address X1 stores a first instruction ("XXX1") followed by a "conditional" jump or branch instruction ("JCC-Y1"). The branch is conditional in that the next instruction to be performed may be either the next sequential instruction ("XXX2") or an instruction at a new address ("Y1"). The processor does not know which branch, or "path," will be taken until JCC-Y1 is executed, i.e., reaches the branch execution unit 150.

Assume now that the branch prediction unit 100 and the next IP logic unit 110 have selected instruction XXX1 to be executed. The processor could wait for XXX1 to move through each stage in the pipeline before processing the next instruction, or JCC-Y1. In this case, the branch execution unit 150 would remain idle while JCC-Y1 moves through the pipeline. To improve the processor's performance, JCC-Y1 is placed into the first stage as soon XXX1 moves into the second stage. As a result, JCC-Y1 will be ready for execution as soon as the branch execution unit 150 is finished with XXX1.

When JCC-Y1 moves into the second stage, however, the processor will not know if XXX2 or YYY1 should be placed into the first stage, because this information is only available after JCC-Y1 has been executed by the branch execution unit 150. Therefore, the branch prediction unit 100 "predicts" which branch of the program will be needed. By way of example, Table I shows the movement of the above instruction sequence through the pipeline shown in

FIG. 1. As can be seen at time 6, the branch prediction unit 100 has predicted that instruction YYY1 will follow JCC-Y1. Note that several instruction "clock" cycles may or may not pass between the time JCC-Y1 moves into the second stage and the time YYY1 is placed into the first stage.

Table I. Program Flow

Time	First Stage	Second Stage	Third Stage	Fourth Stage	Int. Stages	Execution Stage
1	XXX1				. . .	
2	JCC-Y1	XXX1			. . .	
3		JCC-Y1	XXX1		. . .	
4			JCC-Y1	XXX1	. . .	
5				JCC-Y1	. . .	
6	YYY1				. . .	
7	YYY2	YYY1			. . .	
. . .	. . .	. . .	. . .	. . .	. . .	. . .
10	YYY5	YYY4	YYY3	YYY2	. . .	XXX1
11	YYY6	YYY5	YYY4	YYY3	. . .	JCC-Y1
12	XXX2				. . .	
13	XXX3	XXX2			. . .	
14	XXX4	XXX3	XXX2		. . .	
15	XXX5	XXX4	XXX3	XXX2	. . .	

When JCC-Y1 is actually executed at time 11, the branch prediction unit 100 has "mispredicted" and, in fact, XXX2 must be processed next. In this case, instructions YYY1 through YYY6, currently in the pipeline, are discarded and the branch execution unit 150 waits for XXX2 to travel through each pipeline stage before it can be executed. This delay, or mispredicted branch "recovery" time, slows the operation of the processor. Moreover, as the

number of stages in a pipeline increases, the delay caused by each mispredicted path may also increase.

SUMMARY

In accordance with an embodiment of the present invention, a mispredicted path side memory is configured to be coupled to an instruction pipeline stage.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates a known architecture for an instruction pipeline.

FIG. 2 is an instruction pipeline according to an embodiment of the present invention.

FIG. 3 is a flow diagram of a branch misprediction recovery method according to an embodiment of the present invention.

DETAILED DESCRIPTION

An embodiment of the present invention is directed to a mispredicted path side memory for an instruction pipeline in a processor. Referring now in detail to the drawings wherein like parts are designated by like reference numerals throughout, FIG. 2 is an instruction pipeline according to an embodiment of the present invention.

The first stage of the pipeline includes a branch prediction unit 200 and a next IP logic unit 210 that select an instruction to be executed. The next IP logic unit 210 is coupled to an instruction cache 220 through a multiplexing unit 215. The next IP logic unit 210 is also coupled to a mispredicted path side memory 260 through a mispredicted path data line for the second stage ("MP Data (S2)"). The result of the selection performed by the branch prediction unit 200

and the next IP logic unit 210 is passed to the instruction cache 220 through the multiplexing unit 215. According to an embodiment of the present invention, the result is also stored in the mispredicted path side memory 260.

The instruction cache 220 is accessed in the second stage of the pipeline, and the result moves into a third stage unit 230 through another multiplexing unit 225. This result may also be stored in the mispredicted path side memory 260 through a mispredicted path data line for the third stage ("MP Data (S3)"). The instruction moves from the third stage unit 230 to a fourth stage unit 240 through still another multiplexing unit 235, and the result may again be stored in the mispredicted path side memory 260, and the instruction eventually reaches a branch execution unit 250 in the execution stage. As with FIG. 1, the "intermediate stages" shown in FIG. 2 imply that any number of stages may exist in a pipeline.

Note that although FIG. 2 shows, for example, that a result is stored in the mispredicted path side memory 260 using the output of the third stage multiplexing unit 235, the result may instead be sent directly from the third stage unit 230 to the mispredicted path side memory 260. Such an approach, or any other approach, can similarly be used in other pipeline stages.

Also note that as an instruction moves from stage to stage in the pipeline, the information that exists in each stage can be different. That is, for example, the third stage unit 230 may receive information and generate a "result," corresponding to that information, that moves into the fourth stage.

Table II shows the movement of the previously described instruction sequence through the instruction pipeline of FIG. 2. At time 6, the branch prediction unit 200 has predicted that instruction YYY1 will follow JCC-Y1. As before, several instruction "clock" cycles may pass

between the time JCC-Y1 moves into the second stage (time 3) and the time YYY1 is placed into the first stage (time 6).

Table II. Program Flow with Mispredicted Path Side Memory

Time	First Stage	Second Stage	Third Stage	Fourth Stage	Int. Stages	Execution Stage
1	XXX1				. . .	
2	JCC-Y1	XXX1			. . .	
3	XXX2	JCC-Y1	XXX1		. . .	
4	XXX3	XXX2	JCC-Y1	XXX1	. . .	
5	XXX4	XXX3	XXX2	JCC Y 1	. . .	
6	YYY1	(store XXX4)	(store XXX3)	(store XXX2)	. . .	
7	YYY2	YYY1			. . .	
. . .	. . .	. . .	. . .	. . .	. . .	. . .
10	YYY5	YYY4	YYY3	YYY2	. . .	XXX1
11	YYY6	YYY5	YYY4	YYY3	. . .	JCC-Y1
12	XXX5	(restore XXX4)	(restore XXX3)	(restore XXX2)	. . .	
13		XXX5	XXX4	XXX3	. . .	
14			XXX5	XXX3	. . .	
15				XXX5	. . .	

According to an embodiment of the present invention, instructions from the non-predicted branch may be placed into the pipeline during this time. That is, even though the processor has predicted that YYY1 will follow JCC-Y1, the XXX2 instruction is nevertheless placed into the first stage at time 3. Similarly, when XXX2 moves into the second stage, XXX3 is placed into the first stage. According to one embodiment of the present invention, the results from the first,

second and third stages are stored into the mispredicted path side memory 260 as they are generated. According to another embodiment of the present invention, at time 6 the results for XXX2 at the fourth stage, XXX3 at the third stage and XXX4 at the second stage are stored into the mispredicted path side memory 260 all at once.

Tables I and II illustrate that instructions from the non-predicted path may be placed into the pipeline during the time that the stages would otherwise be idle. According to another embodiment of the present invention, YYY1 is actually delayed so that instructions from the non-predicted branch can be executed. That is, the processor "steals" cycles from the predicted, or "main," path because early pipeline stages may have excess bandwidth as compared to the processor's execution capabilities. In such a case, stealing cycles may not greatly reduce performance.

Note that instructions XXX2, XXX3 and XXX4 are executed even though the branch prediction unit 200 has predicted that these instructions will not be needed, and the results of processing these instructions remain stored in the mispredicted path side memory 260 until the associated branch ("JCC-Y1") is executed.

Referring again to Table II, when JCC-Y1 is actually executed at time 11, the branch prediction unit 200 has "mispredicted" and, in fact, XXX2 must be processed next. As a result, instructions YYY1 through YYY6, currently in the pipeline, are discarded.

In this case, however, XXX2 does not need to travel through each pipeline stage before it can be executed. Instead, at time 12 the branch execution unit 250, acting as a mispredicted path side memory control unit, determines that the branch has been mispredicted and sends a signal to the mispredicted path side memory 260 through a read mispredicted path side memory ("read MPSM") control line. This causes the results for XXX2 at the fourth stage, XXX3 at the third

stage and XXX4 at the second stage to be restored from the mispredicted path side memory 260 back into the appropriate pipeline stages. This may be done through restored data ("RP Data") lines between the mispredicted path side memory 260 and the multiplexing units 215, 225, 235. Note that some other device may act as the mispredicted path side memory control unit in place of the branch execution unit 250.

In this way, XXX2 only needs to travel from the fourth (not the first) stage to the branch execution unit 250, saving three instruction clock cycles and improving the processor's performance. Moreover, additional pipeline stages may be added to the processor without increasing the delay caused by a mispredicted path.

The reduced latency achieved in the event of a mispredicted branch may more than offset any reduction in performance caused by cycles that are stolen from the predicted path as described above. There may be, according to one embodiment of the present invention, an optimal number of stages that should be stored in the mispredicted path side memory 260. That is, storing too many stages into the mispredicted path side memory 260 may the delay the execution of correctly predicted paths and decrease the processor's overall performance. The optimal number of stages may depend on, for example, how well the pre-fetch bandwidth is utilized, i.e., whether or not there is free bandwidth to steal. The optimal number of stages may also depend on whether or not the stored recovery information can be quickly used. In other words, how soon after a misprediction can the branch execution unit 250 accept a new stream?

According to one embodiment of the present invention, the mispredicted path side memory 260 is organized as a First-In, First-Out (FIFO) memory. In this case, the misprediction information only exists when the associated branch is in the pipeline. Such an arrangement has the advantage of being relatively simple, but a mispredicted path may need to be re-executed

each time it is encountered. According to another embodiment of the present invention, the mispredicted path side memory 260 is organized as a small cache. This is more complex than a FIFO arrangement, but may prevent multiple executions of a mispredicted path.

The mispredicted path side memory 260 stores information that may also exist in a main cache or a main memory (not shown in FIG. 2). As a result, provisions may be needed to maintain coherence between these devices according to one embodiment of the present invention.

FIG. 3 is a flow diagram of a branch misprediction recovery method according to an embodiment of the present invention. At 310, the processor predicts that a first sequence of instructions, or branch, will be executed and that a second sequence of instructions will not be executed. Instructions from the second sequence are advanced through a plurality of instruction pipeline stages at step 320.

A result of the second sequence is stored from a stage in the pipeline at step 330, such as by being stored into a mispredicted path side memory, and instructions from the first sequence are advanced through the plurality of stages at step 340.

If the prediction is correct at step 350, instructions proceed through the pipeline as predicted, and the information stored in the mispredicted path side memory is not needed. When the prediction is incorrect at step 350, however, the result is restored into a pipeline stage at step 360 to reduce the time needed to recover from the mispredicted path.

Although various embodiments are specifically illustrated and described herein, it will be appreciated that modifications and variations of the present invention are covered by the above teachings and within the purview of the appended claims without departing from the spirit and intended scope of the invention. For example, although a specific mispredicted path side memory and associated control lines were used to illustrate embodiments of the present

invention, it will be appreciated that other implementations will also fall within the scope of the invention. Moreover, the present invention applies to a broad range of pipeline architectures, and is therefore a general approach that includes a broad range of specific implementations. In addition, although software or hardware are described to control certain functions, such functions can be performed using either software, hardware or a combination of software and hardware, as is well known in the art. As is also known, software may be stored, such as in memory, in the form of instructions, including micro-code instructions, adapted to be executed by a processor. As used herein, the phrase "adapted to be executed by a processor" encompasses instructions that need to be translated before being executed by the processor.